

汇编语言程序设计

Assembly Language Programming

第二章 8086指令系统

8086指令系统分成下列六大类：

- ❖ 数据传送指令
- ❖ 算术运算指令
- ❖ 逻辑运算和移位指令
- ❖ 控制转移指令
- ❖ CPU控制指令
- ❖ 串操作指令

汇编指令学习时应注意的问题

- ❏ 寻址方式的多样性
- ❏ 对标志寄存器的影响
- ❏ 两个操作数大小匹配原则
 - ❖ 隐式匹配：两者之中有一个确定即可，CPU 自动匹配
 - ❖ 显式匹配：两者大小都不确定，显式转换。
不确定操作数：im, Mem（变量确定!）

1. 数据传送指令

特点

- ❖ 数据在传送过程中不发生变化
- ❖ 对标志寄存器的内容无任何影响
- ❖ 数据传送的Copy性质。

MOV/XCHG/LEA,LDS,LES/PUSH,POP, PUSHF ,POPF
/XLAT/LAHF,SAHF/...

MOV指令

❏ 格式: MOV Dst, Src

MOV reg/mem/seg, reg/mem/seg/imm

❏ 执行操作: 将源操作数src复制到目的操作数dest,
src不变。

Notice!

- ❏ 目的操作数Dst不能为im, IP, CS
- ❏ 大小要匹配
 - ❖ MOV DL, AX
- ❏ Mem<——>Mem
 - ❖ Mov [1000H], [2000H]
- ❏ 两个段寄存器之间不能直接传送数据；立即数不能直接送入段寄存器中。
 - ❖ MOV DS,ES
 - ❖ MOV SS, 3456H
 - ❖ MOV ES, [1000H]
 - ❖ MOV AX, 1234H
 - ❖ MOV DS,AX

XCHG

❏ 格式: XCHG OPRD1, OPRD2

XCHG reg/mem, reg/mem

❏ 执行操作: **(OPRD1) \leftrightarrow (OPRD2)**

Notice!

- ❑ 不允许使用段寄存器、IP
- ❑ 不能同时为mem
- ❑ 类型要匹配

XLAT

- ❏ 格式: XLAT
- ❏ 功能: $AL \leftarrow DS:(BX) + (AL)$ 所指的单元内容
- ❏ 使用XLAT指令预先应做以下工作
 - ❖ 将表的首地址 (偏移地址) $\rightarrow BX$
 - ❖ 将信息在表中的行号 $\rightarrow AL$
- ❏ 用途: 查表

Notice!

❏ 等价于

- ❖ MOV AH, 0
- ❖ ADD BX, AX
- ❖ MOV AL, [BX]

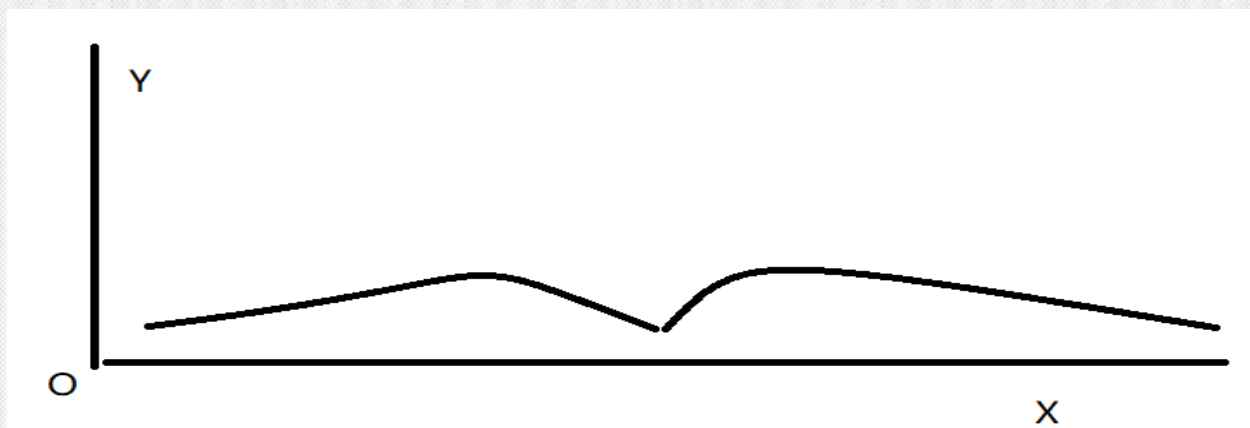
例1

求0~15的平方

- ❖ 将0~15的平方预先放在从T开始的缓存中
- ❖ MOV BX,OFFSET T
- ❖ MOV AL,n
- ❖ XLAT

例2

由输入值得到输出值



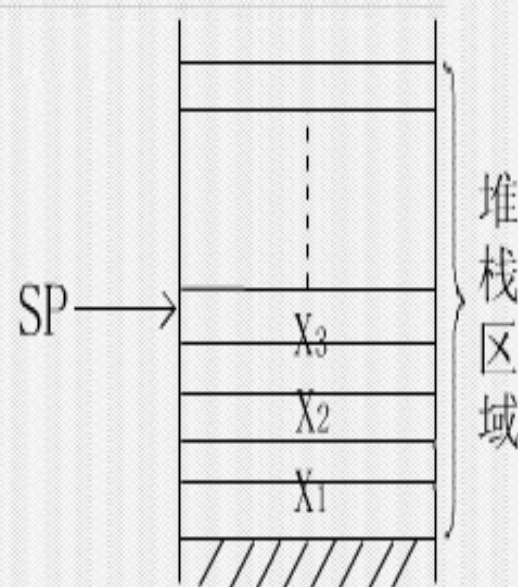
方法

- ❖ 将X和Y的对应关系保存到表格中，可以是一维数组，可保存至ROM
- ❖ 对于输入值，采用二分查找进行计算

堆栈操作指令

堆栈概念 (Stack) :

- ❖ 堆栈是一个特殊的存储区域;
- ❖ 它的一端固定, 另一端浮动;
- ❖ 数据输入输出均在浮动一端 (栈顶) 进行;
- ❖ 按照 “先进后出” 的原理工作。



Why need stack:

- ❖ 函数调用: 断点和局部变量, 参数的保存

PUSH指令

❏ 格式: PUSH Src

❖ PUSH reg16/seg/mem16

❏ 功能:

❖ $(SP)-2 \rightarrow SP$, $(Src) \rightarrow SS:SP$

Notice!

- 必须是字
- PUSH SP尽量不使用

POP指令

❏ 格式: POP Dst

❖ POP reg16/seg/mem16

❏ 功能:

❖ 16位: (SS:SP) \rightarrow Dst, (SP)+2 \rightarrow SP

❖ 说明:

❖ DST不能是CS/IP

❖ 尽量不要使用SP

标志寄存器传送指令

LAHF (Load AH with Flags) 指令

- ❖ 格式: LAHF
- ❖ 功能: F的低8位内容→AH

SAHF (Store AH Into Flags) 指令

- ❖ 格式: SAHF
- ❖ 功能: (AH) →F的低8位


PUSHF (PUSH Flags Into Stack) 指令

- ❖ 格式: PUSHF
- ❖ 功能: PUSH FLAGS

POPF (POP Flags From Stack) 指令

- ❖ 格式: POPF
- ❖ 功能: POP FLAGS (是否影响标志位?)

标志处理指令



{ CLC $CF \leftarrow 0$
 CMC $CF \leftarrow \neg CF$
 STC $CF \leftarrow 1$

{ CLD $DF \leftarrow 0$
 STD $DF \leftarrow 1$

{ CLI $IF \leftarrow 0$
 STI $IF \leftarrow 1$

注意: 只影响本指令指定的标志

地址传送指令LEA

- 格式: LEA Dst, Src
- 功能: $\text{Dst} \leftarrow \text{Src}$ 的偏移地址
- 说明: Dst——16位地址寄存器, Src——Mem
- 例如: LEA BX, BUFFER
LEA SI, [1000H]
LEA DI, [BX+10]

LDS指令

- 格式: LDS Dst, Mem32
- 说明: Dst——16位地址寄存器
- 功能: Mem32的低字→Dst, Mem32的高字→DS
- 例如: LDS BX, [2000H]

LDS BX, BUFFER

LES指令

- 格式: LES Dst, Mem32
- 说明: Dst——16位地址寄存器
- 功能: Mem32的低字→Dst, Mem32的高字→ES
- 例如: LES SI, [1000H]

LES SI, ExtraDATA

8086指令系统分成下列六大类：

- ❖ 数据传送指令
- ❖ 算术运算指令
- ❖ 逻辑运算和移位指令
- ❖ 控制转移指令
- ❖ CPU控制指令
- ❖ 串操作指令

2. 算术运算指令

- ❏ 加法指令
- ❏ 减法指令
- ❏ 乘法指令
- ❏ 除法指令
- ❏ 十进制/BCD码调整指令 (D)

加法指令:ADD

ADD/ADC/INC

ADD指令

- ❖ 格式: `ADD Dst,Src`
- ❖ 功能: $Dst \leftarrow (Dst) + (Src)$
- ❖ 说明:
 - `Dst`——`Reg`, `Mem`;
 - `Src`——`Reg`, `Mem`, `im`
- ❖ 举例:
 - `ADD AX, 10000`
 - `ADD WORD PTR [1000], -1`

加法指令:ADC

❏ ADC指令

- ❖ 格式: `ADC Dst,Src`
- ❖ 功能: $\text{Dst} \leftarrow (\text{Dst}) + (\text{Src}) + \text{CF}$
- ❖ 说明:
 - `Dst`——`Reg`, `Mem`;
 - `Src`——`Reg`, `Mem`, `im`
- ❖ 用途: 用于多字节数相加,不单独使用。
- ❖ 举例:
 - `ADC AX, 10000`
 - `ADC WORD PTR [1000], -1`

多字节数相加

❏ DX= 0002H AX= 0F365H

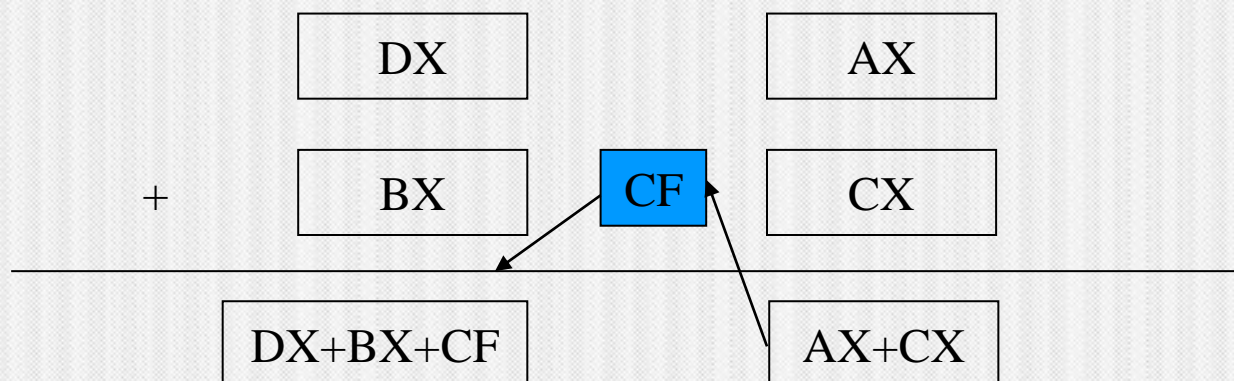
❏ BX= 0005H CX= 0E024H

❏ (1) ADD AX, CX

❖ 执行后, AX= 0D389H CF=1

❏ (2) ADC DX, BX

❖ 执行后, DX= 0008H CF=0



加法指令:INC

INC指令

- ❖ 格式: INC OP
- ❖ 功能: $OP \leftarrow (OP) + 1$
- ❖ 说明: OP——Reg, Mem;
- ❖ 举例:
 - INC BYTE PTR[BX]
 - ADD BYTE PTR[BX],1

Notice!

❏ 除INC指令不影响CF标志外，其他均对条件标志位有影响。

❖ $BX = 0FFFFH$

❖ $ADD\ BX, 1$

● $CF=1\ OF=0\ SF=0\ ZF=1$

❖ $INC\ BX$

● CF不影响 $OF=0, SF=0\ ZF=1$

减法指令: SUB

❏ SUB/SBB/DEC

❏ NEG/CMP

❏ SUB指令

❖ 格式: SUB Dst, Src

❖ 功能: $\text{Dst} \leftarrow (\text{Dst}) - (\text{Src})$

❖ 说明:

● Dst—Reg, Mem; Src—Reg, Mem, im

❖ 举例: SUB AX, -1

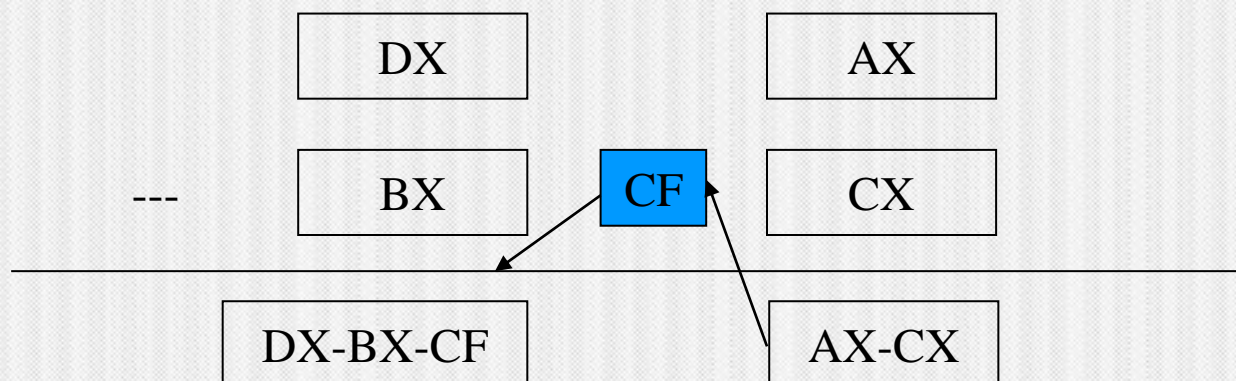
SUB BX, AX

减法指令: SBB

- ❏ 格式: SBB Dst,Src
- ❏ 功能: $\text{Dst} \leftarrow (\text{Dst}) - (\text{Src}) - (\text{CF})$
- ❏ 多字节数相减,不单独使用。
- ❏ 举例:
 - ❖ SBB AX, [1000];
 - ❖ SBB [BP+SI], DL

多字节数相减

- ❏ $DX = 0005H$ $AX = 0001H$
- ❏ $BX = 0005H$ $CX = 0002H$
- ❏ (1) $SUB\ AX, CX$
 - ❖ 执行后, $AX = 0FFFFH$ $CF=1$
- ❏ (2) $SBB\ DX, BX$
 - ❖ 执行后, $DX = 0FFFFH$ $CF=1$



减法指令:DEC

- ❏ DEC指令
- ❏ 格式: DEC OP
- ❏ 功能: $OP \leftarrow (OP) - 1$
- ❏ 说明: OP——reg, mem;不影响CF;
- ❏ 举例:
 - ❖ DEC AX
 - ❖ DEC WORD PTR [1000H]

减法指令: NEG

❏ 格式: NEG OP

❏ 功能: $OP \leftarrow \overline{(OP)} + 1$, 进行求补运算, 即求一个有符号数的相反数

❏ 说明:

- ❖ 以0-OP判断标志位;
- ❖ 仅当(OP)=0时, (CF)=0, 否则CF = 1;
- ❖ 仅当(OP)=-128或-32768时, (OF)=1, 否则OF = 0

❏ 例如:

- ❖ NEG AL ; AL = 0FFH \rightarrow AL = 1;
- ❖ NEG WORD PTR[10H]

减法指令: CMP

- ❏ CMP指令
- ❏ 格式: `CMP Dst, Src`
- ❏ 功能: $(Dst) - (Src)$ 结果的特征状态 $\rightarrow F$, 但 `Dst, Src` 不变
- ❏ 说明:
 - ❖ `Dst`—reg, mem; `Src`—reg, mem, im
- ❏ 举例:
 - ❖ `CMP AL, 60`

Notice!

❏ 除DEC指令不影响CF标志外，其他均对条件标志位有影响。

❖ $(AX) = 0000H, (CX) = 0001H$

❖ SUB AX, CX

- 执行后, $(AX) = 0FFFFH$

- CF=1 OF=0 SF=1 ZF=0

❖ CMP AX, CX

- AX = 0

- CF=1 OF=0 SF=1 ZF=0

❖ DEC AX

- CF不影响 OF=0, SF=1 ZF=0

例3

例：x、y、z均为双精度无符号数，分别存放在地址为X，X+2； Y，Y+2； Z，Z+2的存储单元中，用指令序列实现 $w \leftarrow x+y+24-z$ ，并用W，W+2单元存放w。

```
MOV CX, X
ADD CX, Y
MOV BX, X+2
ADC BX, Y+2 ; x+y
ADD CX, 24
ADC BX, 0 ; x+y+24
SUB CX, Z
SBB BX, Z+2 ; x+y+24-z
MOV W, CX
MOV W+2, BX
```


乘除指令

- ❏ MUL/IMUL
- ❏ DIV/IDIV
- ❏ 严格区分有符号数或者无符号数

乘法指令：MUL

❏ 格式：MUL Src

- ❖ MUL reg8/mem8 ; $AX = AL \times src$
- ❖ MUL reg16/mem16; $DX:AX = AX \times src$

乘法指令：IMUL

- ❏ “IMUL”指令的格式和功能与 “MUL”指令完全相同，只是它用以完成二个带符号数的相乘。
- ❏ 有符号与无符号的差别
 - ❖ $AL = 0FFH$ $BL = 01H$
 - MUL BL
 - $AX = 00FFH$
 - IMUL BL
 - $AX = 0FFFFH$

例: $(AX) = 16A5H$, $(BX) = 0611H$

- (1) **IMUL BL** ; $(AX) \leftarrow (AL) * (BL)$
; $A5 * 11 \Rightarrow 5B * 11 = 060B \Rightarrow F9F5$
; $(AX) = 0F9F5H$
- (2) **MUL BX** ; $(DX, AX) \leftarrow (AX) * (BX)$
; $16A5 * 0611 = 0089\ 5EF5$
; $(DX) = 0089H$ $(AX) = 5EF5H$

除法指令：DIV

格式：DIV SRC

- ❖ DIV reg8/mem8 :
 - $(AL) \leftarrow (AX) / (SRC)$ 的商
 - $(AH) \leftarrow (AX) / (SRC)$ 的余数
- ❖ DIV reg16/mem16 :
 - $(AX) \leftarrow (DX:AX) / (SRC)$ 的商
 - $(DX) \leftarrow (DX:AX) / (SRC)$ 的余数

除法指令: IDIV

- ❏ IDIV指令的格式和功能与 “DIV”指令完全相同，只是它用以完成二个带符号数的相除。
- ❏ 余数与被除数符号一致
 - ❖ $(AX) = -1$,
 - ❖ $(BL) = 2$, IDIV BL , 商 (AL) : 0; 余数 (AH) : -1
 - ❖ $1 \text{ IDIV } -2 = ?$
 - ❖ $-1 \text{ IDIV } -2$

Notice !

- ❏ AX/DX:AX为隐含的被除数寄存器, AL/AX为隐含的商寄存器, AH/DX为隐含的余数寄存器。
- ❏ SRC不能为立即数。
- ❏ 除法溢出错误: 商太大, 若无符号数除法, 当被除数高半部分 \geq 除数时, 溢出错误, 产生0号中断。使用时应该注意。

例4 同位数相除

❏ BL/CL

❏ 无符号数:

❖ MOV AL,BL

❖ MOV AH,0

❖ DIV CL

❏ 有符号数:

❖ MOV AL,BL

❖ MOV AH, 00H/0FFH (怎么办?)

❖ IDIV CL


扩展指令



符号扩展指令

- ❖ **CBW、CWD功能：将符号位进行扩展**
- ❖ **CBW $AL \rightarrow AX$**
- ❖ **CWD $AX \rightarrow (DX, AX)$**

例5: x, y, z, v 均为16位带符号数, 计算 $(v - (x * y + z - 540)) / x$



```
MOV AX, X
IMUL Y                ; x*y
MOV CX, AX
MOV BX, DX
MOV AX, Z
CWD
ADD CX, AX
ADC BX, DX            ; x*y+z
SUB CX, 540           ; ?
SBB BX, 0             ; x*y+z-540
MOV AX, V
CWD
SUB AX, CX
SBB DX, BX            ; v-(x*y+z-540)
IDIV X                ; (v-(x*y+z-540))/x
```